05. 6

# DirectUI™ SDK 使用指南

*DirectUI ver2.0 系列教程*

本文档介绍了 DirectUI 的思想、整体架构与具体的

使用方法，在文档的最后还列举了一个制作案例，使

读者可以快速地掌握 DirectUI 的使用方法。

*本文档版本号：2.20.00　最近修正：2010 年 5 月，创建：2008 年 3*

DirectUI 网站：www.directui.com

公　　司：上海勇进软件有限公司

公司网站：www.uipower.com

# 一、DIRECTUI 整体介绍

## 1.1 DIRECTUI 体系架构介绍

### 1.1.1 DIRECTUI 概念

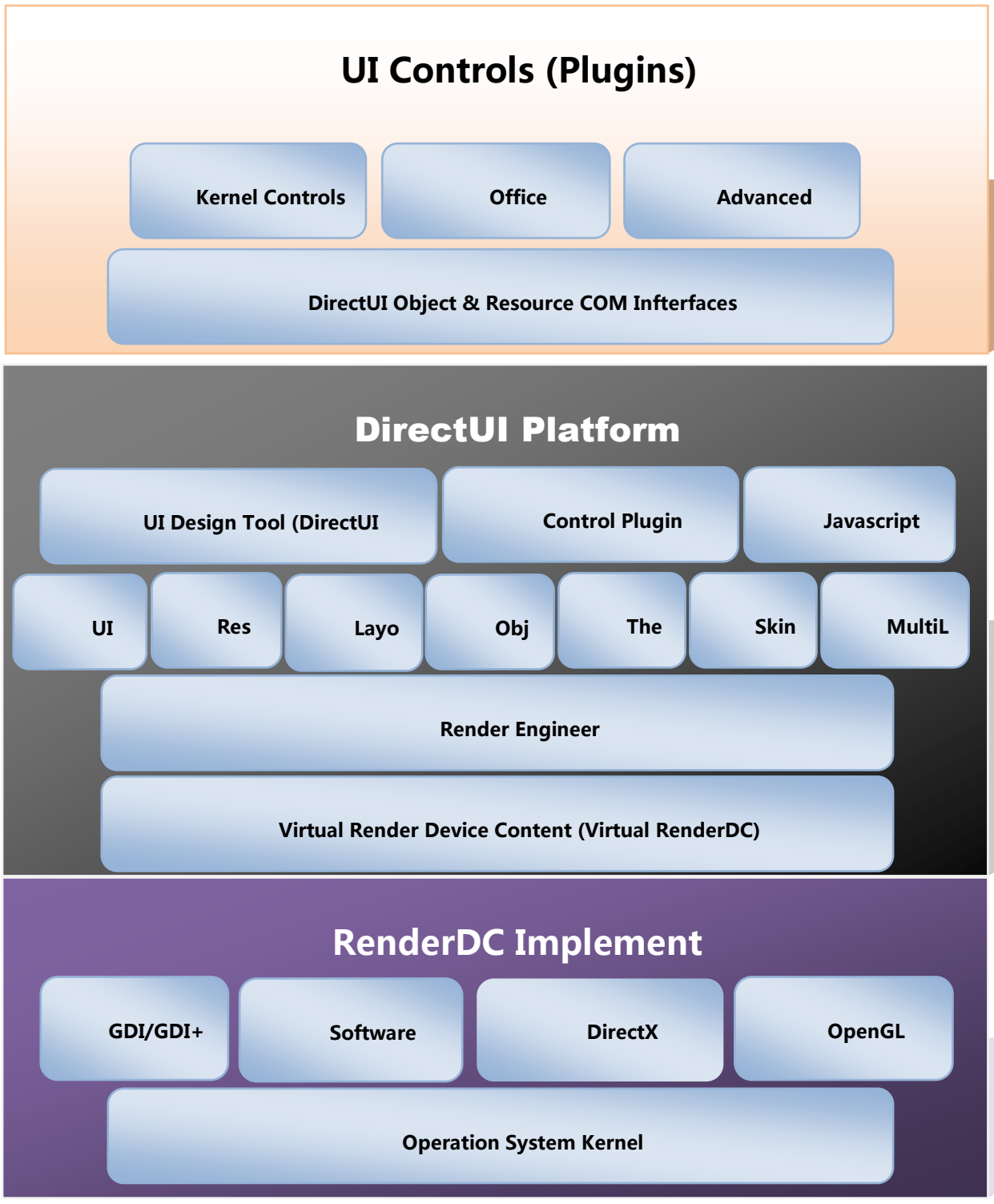
DirectUI 意为直接在对话窗口上绘图，即子窗口不以窗口句柄的形式创建(Windowless)，只是逻辑上的窗口，绘制在对话窗口上。传统的 Windows 界面，是用户窗口(带 Win32 Hwnd 句柄)的层次排列，这种方式有很多局限性。目前的开发环境我们可以很快的使用 win32 控件构造出标准的干净的界面，但是我们同时也会立刻意识到如果用户想让界面更加漂亮，更加酷炫就比较麻烦。DirectUI 界面库使用 XML 来描述界面风格，界面布局，使用脚本语言（如 vbscript，javascript）来实现界面逻辑与业务逻辑的彻底分离。

### 1.1.2 DIRECTUI 主要特性：

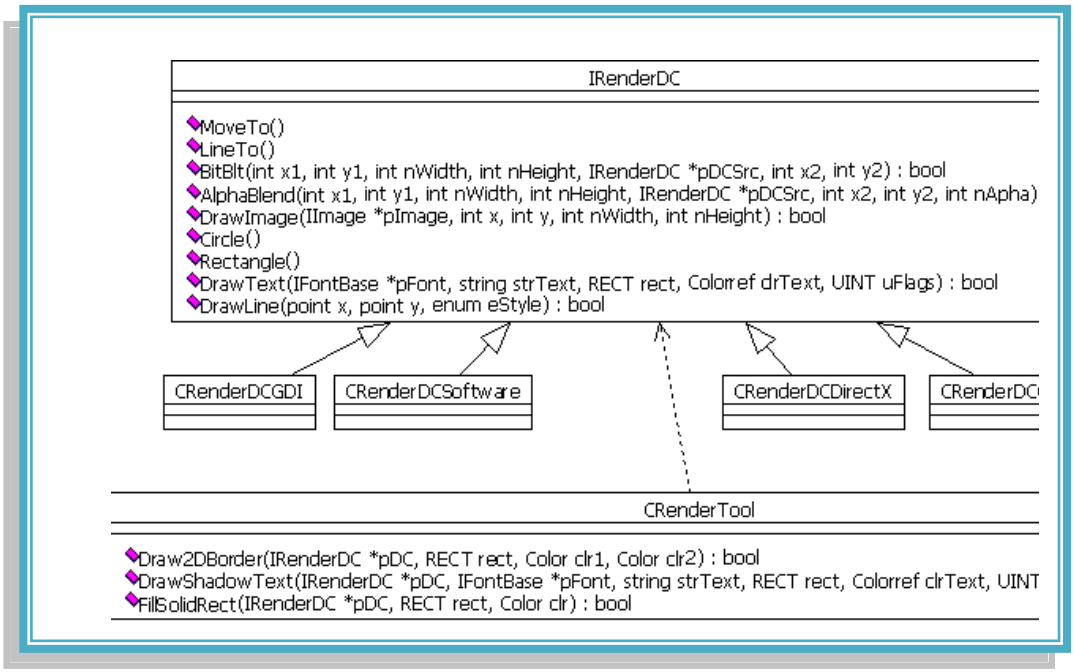
1) 支持皮肤对象的布局；
2) 支持皮肤脚本(JavaScript)控制,可以让界面与逻辑彻底分离；
3) 支持 bmp、png、jpg、gif、tga 等图片格式；
4) 支持多图层 Alpha 混合特效；
5) 提供界面设计工具 DirectUI Builder,支持拖拽式界面设计，让界面开发所见即所得；
6) 开放式开发平台，所有控件均为插件方式管理，支持用户自定义控件开发，与 DirectUI 平台无缝兼容；
7) 支持 Windows 平台所有的开发工具(VC++、VB6、VS.Net、PowerBuilder、Delphi、C++Builder、E 语言)；
8) 支持所有标准控件的换肤；
9) 支持皮肤对象的导出与导入；
10) 支持 Windows 主题导入，让标准界面皮肤的制作简单快捷；

### 1.1.3 DIRECTUI 体系架构

体系架构图如下所示：



- **DirectUI 采用平台+插件的体系架构。**DirectUI 2 系列版本中所有的控件均为插件方式管理。常用的控件被封装在 KernelAll 控件工程中。用户可以使用插件开发向导来开发自己的控件。在 DirectUI 以后的版本中，插件将分几种类型：控件插件、特效插件、图像解码插件、脚本插件等。
- **DirectUI 平台**包含：绘图引擎、UI 逻辑、事件处理、对象布局、脚本控制、共享资源、插件控制器、标准控件换肤引擎、多语种控制器、界面开发可视化工具 DirectUIBuilder 等。
- **DirectUI 的绘图引擎**是最核心的功能模块，其功能强弱、效率高低、内存占用高低等都直接影响到 DirectUI 整体的性能。绘图引擎采用纯虚的图像设备上下文来对各种操作系统进行全面的支持。Virtual RenderDC 将各种图像与文字的处理设计成各种类，并将每个类的方法设计成纯虚函数。如果要支持例如 DirectX 绘图引擎，只需要将那些纯虚的各种类与接口实现即可。所以通过该机制 DirectUI 实现了跨平台。其类图大致如下：




- **UI Logic**：负责处理所有控件共用的各种逻辑。
- **Share Resource**：DirectUI 将所有的图片、字体、图像选区、文本样式、字体、光标等称之为资源，并将这些资源进行计数引用式管理，从而很好地避免数据的冗余与内存的最小化。
- **Object Layout Manager**:DirectUI 中所有的对象均有布局的概念，并且布局器支持 2 种以上的布局：父子布局、表格布局、绝对布局等。有了该项功能后用户不再需要程序中对各种控件进行布局了。对象布局信息被包含在皮肤方案文件中，所以更换一套皮肤方案就能更改一种布局。
- **Control Objects**：DirectUI 中将所有的控件统称为对象，在平台层将每个控件共用的属性与行为进行了基类的封装。

- **DirectUI Builder**：DirectUI 提供了所见即所得的界面开发工具 DirectUIBuilder。用户通过该工具可以对界面进行可视化的设计：支持拖拽式的控件布局，支持控件的拷贝与粘贴，支持控件的导出与导入，对大团队开发进行了强有力的支持。
- **Control Plugin Manager**：DirectUI 中所有的控件均为插件。在平台层有对所有类型插件进行管理的机制，方便用户编写各种类型的插件，提供控件开发的控件型插件向导。用户通过该向导可以一步一步地将控件与平台接口的各种属性进行确定，最终生成一个可供 DirectUI Builder 无缝兼容的新控件。
- **Javascript Control**：DirectUI 通过 Javascript 的控制来真正实现界面逻辑与业务逻辑的彻底分离。Javascript 语法简单很容易被 C++程序员掌握，DirectUI 采用 COM 技术对 Javascript 的功能对象进行了扩充。将 DirectUI 中所有的控件接口自动添加进 Javascript 引擎，从而可以在 Javascript 中方便地访问各种控件接口中的方法。脚本功能被包含在 DirectUIBuilder 中，用户填写的脚本代码被包含在皮肤方案文件 SKN 中，所以更换了 SKN 文件，脚本的界面逻辑控制代码也随即一起更换了。
- **Theme**：DirectUI 支持多主题功能 通过主题用户可以节约大部分的控件属性的重复设置工作。一般的软件界面项目中 80%的控件均为相同样式的，只有很少的大概占 20%的控件才需要特殊的风格设置，所以主题是解决该 80%控件重复设置的问题，同时主题还可以跨项目跨软件使用，其重用程度相当于 Windows 的主题文件。
- **SkinCtrl：** 在一套软件界面中，从总体上来说分标准控件的换肤与自定义控件界面定义。SkinCtrl 就用来对 Windows 标准控件进行换肤的模块。即使在用户的程序中没有用到一个 Windows 标准控件，但还是需要对 SkinCtrl 模块的界面属性进行设置。因为在所有的程序中几乎都有 MessageBox、打开文件对话框、颜色选择对话框、目录选择对话框等 Windows 系统窗口，这些窗口都属于标准控件的范畴。
- **MultiLang:** DirectUI 支持多语种功能，用户只要使用 DirectUIBuilder 中多语种管理窗口即可实现多国语种的管理与动态切换语种的功能。
- **UI Controls(Plugins)**：目前 2.0 版本中控件分成 4 大系列：核心控件组、办公类软件控件组、高级类控件组、工业类软件控件组。这些控件都是以 DirectUI 的插件形式存在，可以单独加载与卸载。给程序发布提供了灵活的解决方案。

## 1.2 DIRECTUI 界面方案制作介绍

在 DirectUI 应用程序界面开发模式中，工作主要集中在 DirectUI 界面方案制作与程序调用。而界面方案制作占整个开发工作量的 90%。界面方案的制作步骤为美术设计->图片切图->DirectUIBuilder 界面方案文件制作。

- **美术设计**：美术设计师根据程序开发需求进行界面的用户交互体验的调查与设计，迭代多次后确定新界面的交互模式与习惯。接下来设计师即进行整体的效果设计，绘制大体的低保真的整体效果图，经过多次迭代后确定新界面的整体效果，并生成高保真的效果图。同时需要生成一系列界面窗口的控件坐标信息，供接下来的界面方案文件的制作提供坐标位置信息。也供该项目测试人员对 UI 的完稿进行测评的依据。

- **图片切图**：设计师首先要将效果图中涉及到的所有控件进行各种状态的设计，比如按钮控件具有 5 个状态(正常、高亮、按下、禁用、焦点)，设计师需要对其进行 5 个状态的设计。当把所有的控件状态设计完成后即可进行图片的切割。切割后的小图片可以集中放在一个图片文件中也可以按控件类型或名称进行另存。DirectUIBuilder 可以加载一张图片文件并允许用户选取其中的一个位置作为某个控件的贴图元素，所以以上 2 种控件状态图的存放均受 DirectUIBuilder 的支持。目前支持格式为 Bmp、Png、Tga 等。

- **DirectUIBuilder 界面方案文件制作**：DirectUI SDK 中提供了制作界面方案的工具 DirectUI Builder。以往界面库多数让用户直接修改 XML 文件进行配置。该种方式制作界面比较花费时间与精力，而且不太方便以后的维护。特别是对维护 XML 的人员要求很高 需要熟悉 XML 的语法。对于没有编程基础的最终用户来说 这种配置界面的方式显然是不合适的。DirectUI 的实质也是将所有的界面元素与界面逻辑保存为 XML，并在窗口新建时使用 XML 相应节点中的数据进行窗口与控件的创建、布局与贴图，并进行相应的事件处理和界面逻辑的执行。DirectUI Builder 使得界面的配置工作相对各种用户来说变得透明，用户不再需要学习 XML 语法与各种节点特定的含义，只需要通过鼠标进行所见即所得的拖拽即可形成界面方案。该工具还可以提供给最终用户进行界面的定制工作，可以大大提高用户软件的友好性与易用性。

DirectUIBuilder 生成的文件有 .dsln、.dui、.skn、.dexport、.dtheme。其中.dui,.skn 为成果性文件，可供程序来调用，其他类型的文件是用户管理界面方案制作过程的文件。.dui 文件为界面元素的集合，里面包含了界面元素的层次关系，与界面元素的基本属性信息。.skn 文件为界面元素风格属性的集合，里面包含了界面元素的布局、图像选区、文字风格、光标等与外观风格有关的信息。有关这些类型图片文件的详细说明见后续章节的描述。

## 1.3 DIRECTUI 程序调用介绍

- DirectUI 支持的开发工具相对比较广泛，可以支持以下开发工具：
  - Visual C++ MFC (6.0 、7.x、8.0、9.0、10.0)
  - Visual C++.Net
  - Visual Basic 6
  - Visual Basic.Net
  - Visual C#.net ( 1.0、2.0、3.0、4.0 )
  - Delphi (4.0、5.x、6.x、7.x、8.x)
  - C++ Builder(4.0、5.x、6.x、7.x、8.x)
  - PowerBuilder ( 6.x、7.0、8.0、9.0、10.0、11.0 )
  - Visual Foxpro

- DirectUI 是基于 COM 模式的，目前有 2 种调用方式：COM 无注册模式与 COM 注册模式，COM 无注册模式可以不需要用户进行 Regsvr32.exe 的注册即可运行。该方式使用起来类似于规则 DLL。目前在 Visual C++与 C++ Builder 2 个开发工具中给与支持。DirectUI 在默认情况均支持标准的 COM 注册调用方式。COM 注册调用方式，需要用户在第一次使用 DirectUI 时候使用 Regsvr32.exe。

### 1.3.1 在 VISUAL C++ (MFC)中的使用（VISUAL STUDIO 2003 …）

- **准备工作**：在应用程序工程属性页的 C/C++ - 常规 - 附加包含目录里添加 “DirectUI SDK”的绝对或相对当前工程的相对路径。
- **界面加载**：完成以下几步的操作后，程序即能在启动时加载了 DirectUI 界面方案。

1. 在 StdAfx.h 的末尾添加：#include “DirectUI.h”
2. 在 CxxxApp 的头文件.h 中添加 IDUIRes *m_pDUIRes;
3. 在 CxxxApp 的头文件的末尾添加 extern CxxxApp theApp;
4. 在 CxxxApp::InitInstance()的第一行添加：m_pDUIRes = OpenSkin(_T("xxx.dui"),_T("Default.skn"), TRUE);

- **窗口关联**：应用程序中一个窗口对应于界面方案中的一个 DirectUI 对象，要让窗口与某 DirectUI 对象（假设该对象名称：mainWnd）进行关联，需要在窗口所在类的头文件中添加如下声明：

```
IDirectUI    *m_pDirectUI；
```

同时在窗口的 OnCreate 事件中写上

程序创建窗口后就会自动创建 DirectUI 对象中的所有的控件，运行效果与 DirectUIBuilder 中预览的效果完全一致。IDirectUI

```
m_pDirectUI =
    (IDirectUI*)theApp.m_pDUIRes->CreateDirectUI(_T("mainWnd"),HandleToL
    ong(m_hWnd));
```

对象所关联的属性与该窗口相关。例如可以设置其玻璃特效、最小尺寸、最大尺寸、弹出风格等属性。

- **变量关联**：程序中需要操作的控件元素都需要获得该控件的对象，通过调用该控件的方法来对该控件进行操作。获取控件对象的实例

```
theApp.m_pDUIRes->GetResObject(DUIOBJTYPE_PLUGIN,控件对象的名
    称,DirectUI 对象的指针,VARIANT_TRUE);
```

例如要获得按钮(DirectUI 控件对象的名称：BtnClose)的指针：

```
ICmdButton*pBtnClose =
    (ICmdButton*)theApp.m_pDUIRes->GetResObject(DUIOBJTYPE_PLUGIN,_T
    ("BtnClose"),m_pDirectUI,VARIANT_TRUE);
```

- **事件处理**：DirectUI 控件的事件响应均采用 Windows 自定义消息来处理，

```
ON_MESSAGE(DirectUI 预定义的消息 ID，该窗口类定义的该消息处理函数)
```

例如要响应按钮控件的鼠标抬起事件，可以如下声明消息：

```
ON_MESSAGE(DUISM_LBUTTONUP,OnSMLButtonUp)
```

由于每个控件的消息处理均有所不同，需要查 DSDN 开发文档中心。

- **动态换肤**：DirectUI 支持动态切换皮肤，只需要在程序任何地方调用

```
ChangeSkin（skn 文件的路径）;
```

即可实现换肤。

- **退出清理**：在程序退出时，需要对 DirectUI 所占的资源进行清理。在 CxxxApp:: ExitInstance()的第一行添加：

```
FreeSkin();
```

### 1.3.2 在 C++ BUILDER 中的使用

- **准备工作：**修改工程引用目录,将 DirectUI SDK 添加到引入目录中，并将 KDUI32.lib 添加到包含库中，将 KDUI32.dll 与 DirectUICom.dll 拷贝到程序运行目录下。

- **界面加载：**建立一个资源管理类和 DirectUI 资源对象,

```
IDUIRes            *g_pDUIRes = NULL;
class ResManager
{
public:
   ResManager()
   {
            g_pDUIRes = OpenSkin(L"360Demo.dui", L"Default.skn", FALSE, FALSE );
   }
   ~ResManager()
   {
            FreeSkin();
   }
};
```

g_pDUIRes 为 DirectUI 资源对象，通过该对象我们可以提取皮肤中的资源

OpenSkin 为打开皮肤方法，参数分别为 Dui 文件路径，Skn 文件路径，是否使用 SkinCtrl，是否自定添加内部引用计数。

```
在 WinMain 外申明此对象。
ResManager resManager;
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int){ … }
```

FreeSkin 为皮肤资源释放方法，在程序退出时调用。

1. **窗口关联：**创建一个 TForm 窗口，在成员变量中申明一个 DirectUI 对象接口，并修改窗口的创建方法将其进行绑定。

```
class TForm2 : public   TForm
{
__published:// IDE-managed Components
        void __fastcall FormCreate(TObject *Sender);
private:// User declarations
        IDirectUI       *m_pDirectUI;
BEGIN_MESSAGE_MAP
END_MESSAGE_MAP(TForm)
}
void __fastcall TForm2::FormCreate(TObject *Sender)
{
        g_pDUIRes->CreateDirectUI(L"HealCheck", HandleToLong(Handle),
    (IDispatch**)&m_pDirectUI);
}
```

- **变量关联**：程序中需要操作的控件元素都需要获得该控件的对象，通过调用该控件的方法来对该控件进行操作。

```
#define DIRECTUI_GETCONTROL(objname, obj) \
        (IDUIControlBase*)g_pDUIRes->GetResObject(DUIOBJTYPE_PLUGIN,obj
    name,m_pDirectUI,VARIANT_TRUE, (IDUIObj**)obj)
```

DIRECTUI_GETCONTROL 是一个绑定宏，方便我们对控件进行绑定，该定义如下

```
ICmdButton*m_pBtnClose
```

例如要关联一个按钮控件，我们需要首先申明一个按钮对象指针。

```
DIRECTUI_GETCONTROL(L"BtnClose", &m_pBtnClose);
```

在 FormCreate 中绑定该控件接口

- **事件处理**：声明消息接收函数

```
    void __fastcall OnSMLButtonUp(TMessage message);
    BEGIN_MESSAGE_MAP
        MESSAGE_HANDLER(DUISM_LBUTTONUP, TMessage, OnSMLButtonUp)
    END_MESSAGE_MAP(TForm)
```

```
void __fastcall TForm1::OnSMLButtonUp(TMessage message)
{
    IDUIControlBase *pControlBase = (IDUIControlBase*)message.WParam;
    if (pControlBase)
    {
        if (pControlBase == (IDUIControlBase*)m_pBtnClose)
        {
            this->Close();
        }
}
```

DUISM_LBUTTONUP 为 DirectUI 控件点击消息

- **退出清理**：资源管理类在析构时自动调用 FreeSkin();

### 1.3.3 在 DELPHI 中的使用

### 1.3.4 在 VISUAL BASIC 6 中的使用

### 1.3.5 在 VISUAL C# 中的使用

- **准备工作**：将 DirectUICom.dll、KernelAll.dll、OfficeAll.dll、AdvancedAll.dll、IndustryAll.dll 添加引用。

  将皮肤文件（xxx.dui、xxx.skn 以及 xxx.xml）拷到 Bin 目录。

- **界面加载**：完成以下几步的操作后，程序即能在启动时加载了 DirectUI 界面方案。

1. 在 program.cs 中声明 DUIManagerClass 对象和 DUIRes 对象。
2. 新建 DUIManagerClass 对象，并获得 DUIRes 对象。

3. 加载皮肤文件。
   相关代码如下：

```
        public static DirectUIComLib.DUIManagerClass ComDUIManager = null;
        public static DirectUIComLib.DUIRes DUIResource = null;

        [STAThread]
        static void Main()
        {
            ComDUIManager = new DUIManagerClass();
            ComDUIManager.SetAutoReleaseMode(false);
            DUIResource = (DUIRes)ComDUIManager.GetResource();
            bool bResult = DUIResource.OpenSkin("xxx.dui",
    "Default.skn",false);

//              Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainFrame());
        }
```

- **窗口关联**：应用程序中一个窗口对应于界面方案中的一个 DirectUI 对象，要让窗口与某 DirectUI 对象（假设该对象名称：mainWnd）进行关联：

```
private DirectUIComLib.DirectUI oDirectUI = null;
```

同时在窗口的 Load 事件中写上

```
oDirectUI =
    (DirectUIComLib.DirectUI)Program.DUIResource.CreateDirectUI("mainWnd
    ", (int)this.Handle);
```

程序创建窗口后就会自动创建 DirectUI 对象中的所有的控件，运行效果与 DirectUIBuilder 中预览的效果完全一致。

- **变量关联**：程序中需要操作的控件元素都需要获得该控件的对象，通过调用该控件的方法来对该控件进行操作。获取控件对象的实例

```
Program.DUIResource.GetResObject(DirectUIComLib.DUIObjType.DUIOBJTYPE_
    PLUGIN, 控件名称, (DirectUIComLib.SkinObjResBaseCom)oDirectUI, true);
```

例如要获得按钮(DirectUI 控件对象的名称：BtnClose)的指针：

```
BtnClose =
    (CmdButton)Program.DUIResource.GetResObject(DirectUIComLib.DUIObjT
    ype.DUIOBJTYPE_PLUGIN, "BtnClose",
    (DirectUIComLib.SkinObjResBaseCom)oDirectUI, true);
```

- **事件处理**：DirectUI 控件的事件响应均采用 Windows 自定义消息来处理，

例如要响应按钮控件的鼠标抬起事件（DUISM_LBUTTONUP），可以窗口过程中处理：

```
        protected override void DefWndProc(ref Message m)
        {
            switch (m.Msg)
            {
                case (int)DirectUIComLib.DUIMsgID.DUISM_LBUTTONUP:
                    {
                        DirectUIComLib.DUIObj duiControl =
    (DirectUIComLib.DUIObj)(Marshal.GetObjectForIUnknown(m.WParam));
                        if (duiControl == BtnClose)
                        {
                            Application.Exit();
                        }
                    }
                    break;

                    default:
                    base.DefWndProc(ref m);
                    break;
}
}
```

由于每个控件的消息处理均有所不同，需要查 DSDN 开发文档中心。

### 1.3.6 在 POWERBUILDER 中的使用

# 二、DIRECTUIBUILDER 介绍

## 2.1 整体介绍

DirectUIBuilder 在界面开发中起着关键性的作用。对其掌握的程度直接影响到界面制作的速度与界面运行时的效率与内存占用率。界面库是否易用往往以类似于这样的工具作为第一评价标准。

## 2.1.1 DIRECTUIBUILDER 自身的主要界面元素有：

### 主界面


### 菜单区：

### 文件菜单：

1、 新建、打开、保存与关闭解决方案 dsln；

2、 新建与导入 DUI 文件；

3、 新建与导入 Skn 文件；

### 编辑菜单：

1、 可以对选中的控件对象进行拷贝、粘贴、剪切、删除操作；

2、 可以对当前的操作进行连续的撤销与重做。

## 视图菜单：

可以显示与隐藏各种工具栏、状态栏、解决方案、工程区、资源区、属性区、工具箱区与样式区。

## 工具菜单：

可以启动多语种管理、设置皮肤密钥、输出皮肤、主题管理器

## 帮助菜单：

弹出关于 DirectUIBuilder 的对话窗口。

## 工具栏区：

## 主工具栏：


包含新建、打开与保存解决方案、撤销与重做、剪切、拷贝与粘贴、ZOrder 坐标调整、DirectUI 对象创建、DirectUI 对象最小化设置、还原设置、Help 属性设置、关于 DirectUI。

## 对齐工具栏：


可以对选中的一个或多个对象进行各种对齐属性的设置。

1、 所有选中靠左对齐、靠右对齐、靠上对齐、靠下对齐；

2、 所有选中控件的纵向与横向的居中布局；

3、 所有选中控件的横向与纵向的平均分布；

4、 所有选中控件的宽度、高度、大小完全保持一致。

## 布局工具栏：


可以对选中的对象进行快捷的布局。从左到右的图标的含义分别是： 左上、右上、左下、右下、左边、右边、顶边、底边、充满、居中。

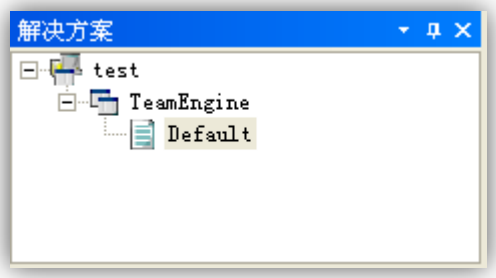
## 语种选择工具栏：


可以根据当前界面解决方案中设置的语种个数，列举出所有的语种供选择，选中一个语种后当前就使用该语种进行文字的设置。

## 主题选择工具栏：


列举出当前 DirectUIBuilder 环境设置中设置的所有主题的名称。选中一个主题后，当前的 SKN 就使用该主题。当界面解决方案有多个 SKN 同时打开时，SKN 之间的切换会导致该主题选择列表的当前主题的动态变换。即一个 SKN 文件对应一个主题。多个 SKN 文件可以使用相同主题。

## 解决方案区



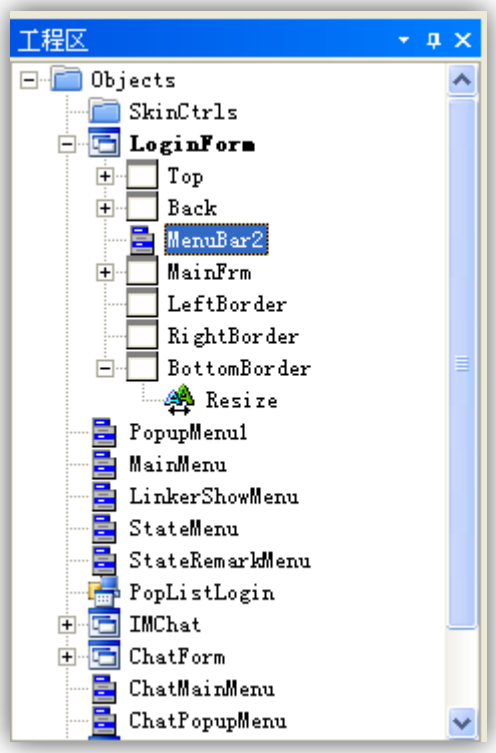
- 该区显示解决方案.dsln 的树状结构，包括.dui、.skn、.dtheme.

```
--dsln
--dui(模块)—(相当于 visualstudio 中的.dsp)
   --skn1
   --skn2
--dui(模块)—(相当于 visualstudio 中的.dsp)
   --skn1
   --skn2
```

树状结构如下：

- 用户可以右键点击 dui 节点弹出菜单，可以增加、移除 dui。
- 用户可以右键点击 skn 节点弹出菜单，可以增加、移除 skn。
- 用户可以点击该区标题的图钉按钮进行该区的固定与停靠。也可以点击关闭按钮隐藏该区，隐藏后可以通过工具菜单再次显示该区。

## 工程区



- 工程区显示的是 dui 文件的内容。描述了一个程序中所有的窗口(DirectUI 对象)和窗口中所包含的控件名称（Controls）。DirectUI 对象与控件、控件与控件之间都是树状结构，即他们之间属于父与子的关系，层层迭代形成一个复杂的界面控件组合。在工程 dui 文件第一次打开时，DirectUI 对象左边没有“+”号，代表其在刚加载进来后并没有一次性将所有的该对象里面所有的对象都读入进来。这样的好处是可以大大提高打开 dui 文件时的加载速度与内存占用。

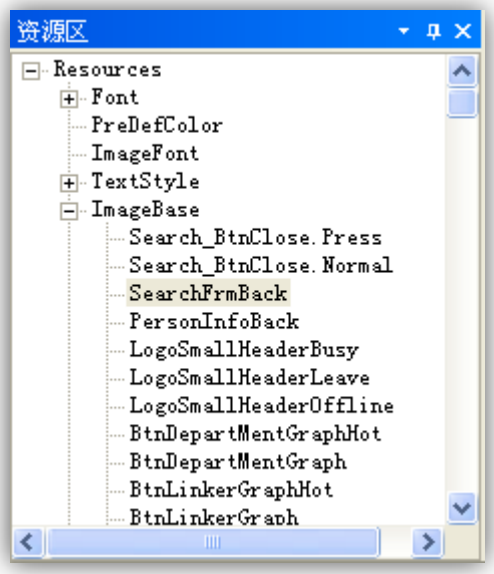
  树状结构如下：

```
-mainframe (DirectUI 对象)
    -background (UIForm 对象)
        -button1
        -button2
    -bottom (UIForm 对象)
        -static1
        -static2
-aboutform (DirectUI 对象)
    -back (UIForm 对象)
        -top (UIForm 对象)
        -middle (UIForm 对象)
```

- 用户可以单击 DirectUI 对象来预览其界面。此时 DirectUIBuilder 将读取该对象下所有子控件，递归读取直到最后。所以用户点击后 DirectUI 对象左边会变成“+”号。代表其子对象已经全部读取进来。当用户在工作区直接点击某个对象后，控件树将自动展开并定位到当前选中对象的节点。

- 用户可以点击该区标题的图钉按钮进行该区的固定与停靠。也可以点击关闭按钮隐藏该区，隐藏后可以通过工具菜单再次显示该区。

## 资源区

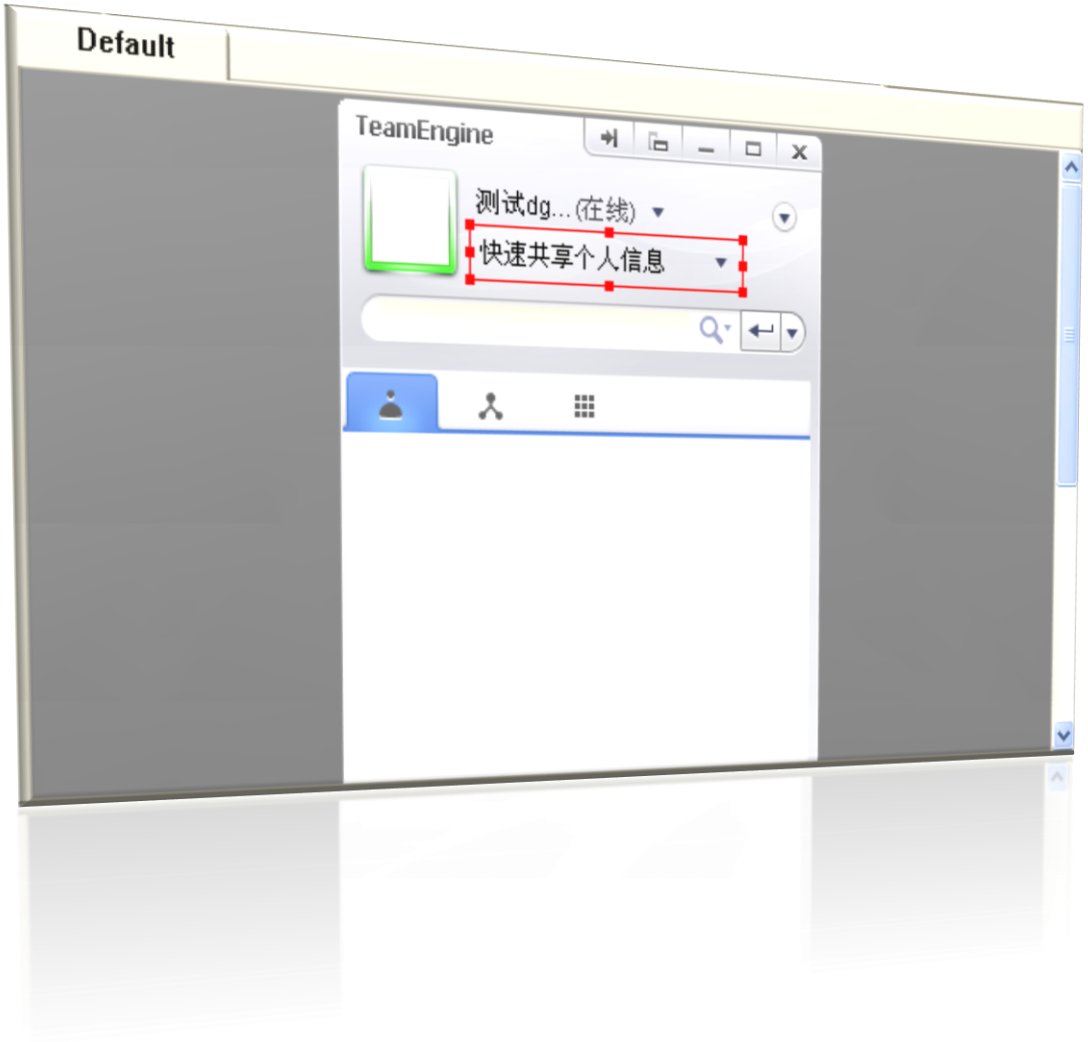



这个区将当前 SKN 文件中用到的所有的共享资源( Image、ImageSection、TextStyle、Font 等 )以分组的方式进行列举。

```
-Font
   -font1
   -font2
-Image
   -image1
   -image2
```

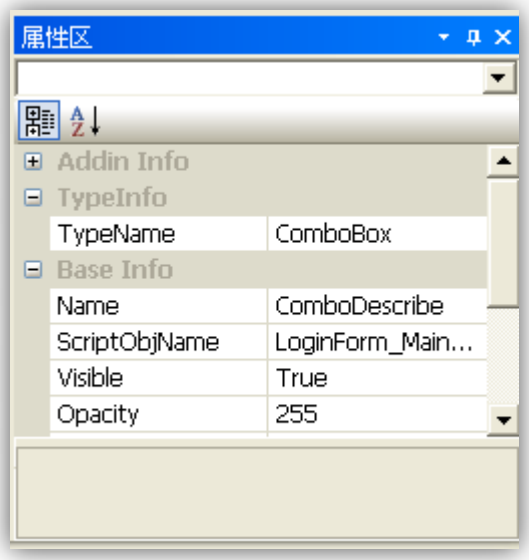
分组结构如下：

## 视图工作区




- 该区域是 DirectUIBuilder 最重要的部分，集界面预览、控件拖拽、复制粘贴、控件停靠、控件对齐等功能一身的工作区。该部分功能是否易用与用户对其掌握程度直接影响到界面制作的效率。
- 控件的外围有红框代表该控件在选中状态，用户可以通过拉伸红框的四个边来对控件进行大小的调整，也可以移动红框来对控件的位置进行调整。用户既可以单选一个控件进行操作，也可以通过按住 Shift 键来多选几个控件进行操作。当控件处在选中状态时，可以按方向键对控件的位置进行微调。也可以通过按住 Shift 键来对单个选中的控件进行大小的微调。
- 该区域的上方有一个 Tab 控件，一个 Tab 页面对应一个 SKN。用户可以通过切换 Tab 页对 SKN 进行切换。

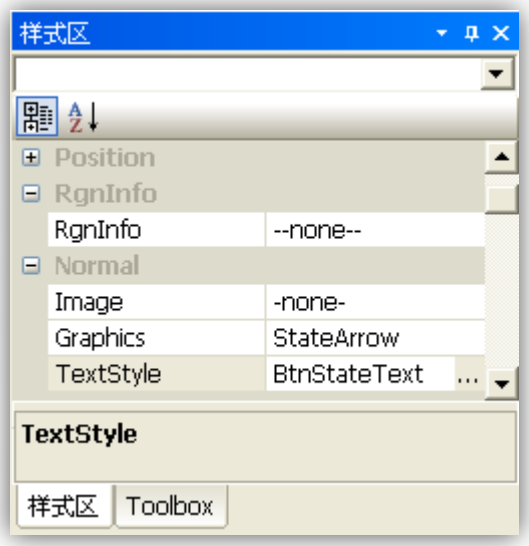
## 属性区




- 每个控件的属性均在属性区进行展示。
- 常用的操作有属性分类节点的展开与收起。属性分类节点的展开与否的状态被记录在 DUI 文件中。所以用户打开 DUI 文件后可以重现上次操作的状态。对于某些控件具有复杂属性分类并属性比较多的情况下，该功能可以提供用户编辑属性的效率。
- 属性区的值与 DUI 文件对应。
- 用户可以点击该区标题的图钉按钮进行该区的固定与停靠。也可以点击关闭按钮隐藏该区，隐藏后可以通过工具菜单再次显示该区。

## 样式区

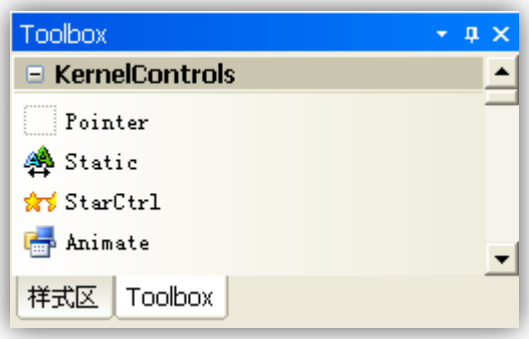
截屏如下：




- 每个控件的图像选区、文本风格、颜色等设置都在样式区进行操作。
- 其分类节点的展开与否的功能与属性区一样，其状态保存在 SKN 文件中。
- 样式区的属性与 SKN 文件对应。
- 用户可以点击该区标题的图钉按钮进行该区的固定与停靠。也可以点击关闭按钮隐藏该区，隐藏后可以通过工具菜单再次显示该区。

## 控件工具箱区

截屏如下：

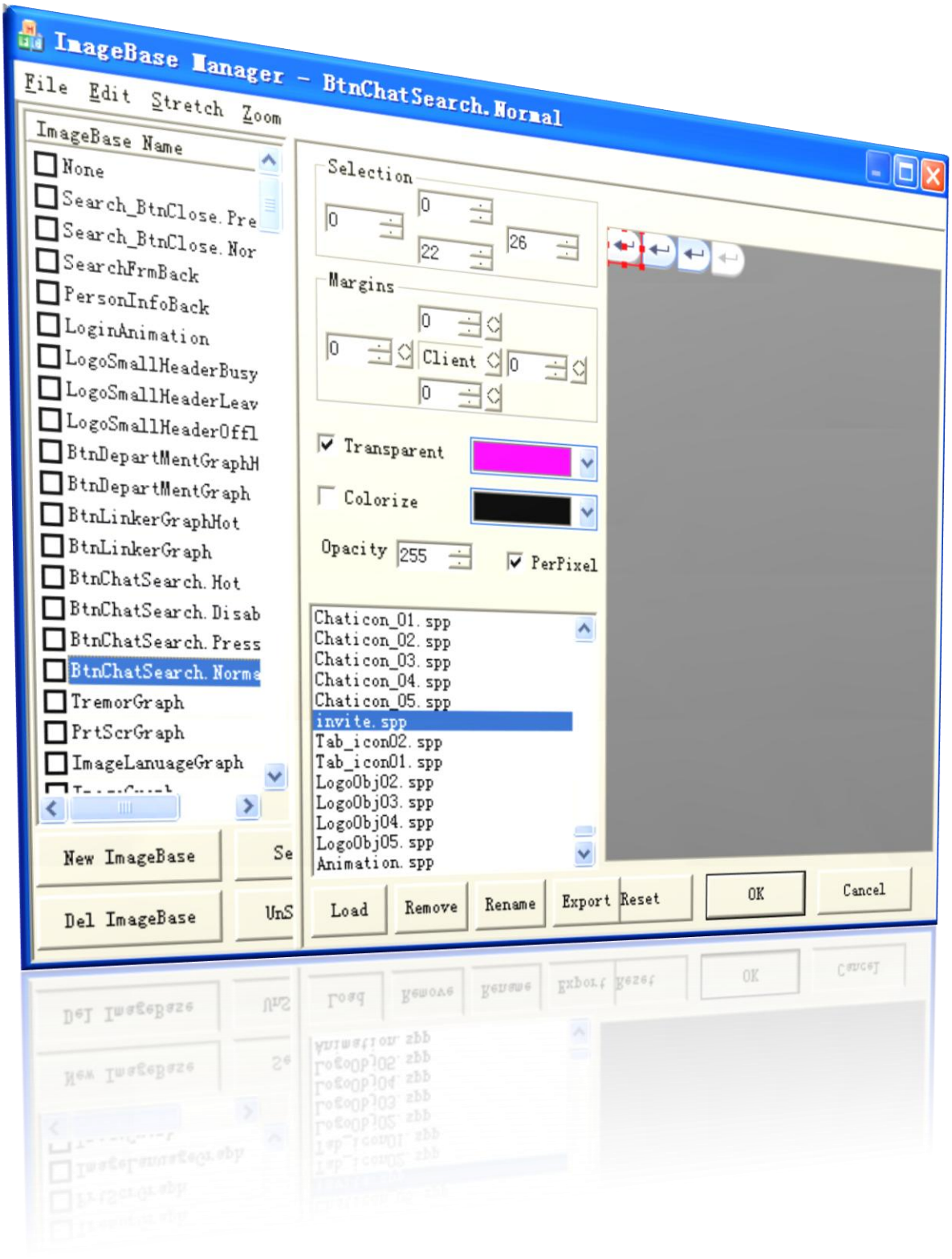



- DirectUI 控件采用插件方式开发与管理。用户可以编写自己的 DirectUI 控件。也可以建立自己的控件分类。工具箱中的每一个控件均有控件的图标、控件的名称、控件的分类、控件拖放时的光标。用户在开发控件需要将这些信息告诉 DirectUIBuilder。

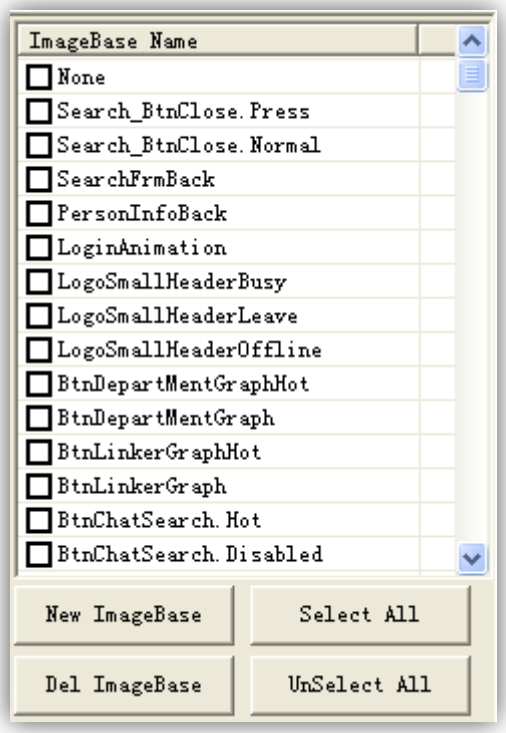
- 目前 DirectUIBuilder 中的自带控件分以下四类：Kernel 控件组、Office 控件组、Advanced 控件组、工业控件组。
- 用户可以点击该区标题的图钉按钮进行该区的固定与停靠。也可以点击关闭按钮隐藏该区，隐藏后可以通过工具菜单再次显示该区。

## 图像选区界面

该界面分 2 大部分：ImageBase 管理区与图像选区编辑区。




IMAGEBASE 管理区

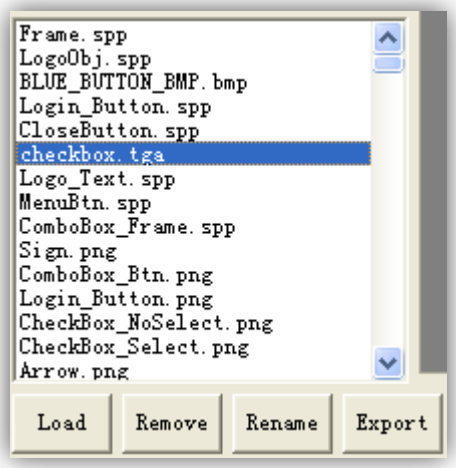



- ImageBase 区管理当前 SKN 文件中所有的 ImageBase，用户可以对其进行创建、编辑与删除等操作。
- ImageBase 的命名不能重复。
- ImageBase 的命名使用英文或数字的组合，不能使用特殊字符。
- ImageBase 的命名尽量做到容易读懂。
- 通过列表前面的复选框，用户部分选择或全选，进行删除操作。

## 图像选区编辑区

ImageBase 对象包含选择哪张图片，选中该图片的哪部分区域，选中区域图片的四边的拉伸与平铺属性，该区域图片的透明色等信息，所以在该区域包含以下部分：

## 图像列表：

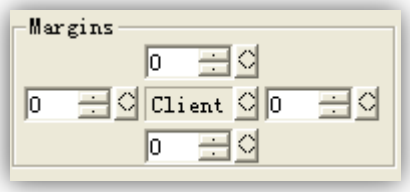



- 在该列表中列举了当前 SKN 文件中所有的图片文件。
- 如果要设置的图片不再当前的 SKN 文件中，用户可以从外部选择一张图片进来。
- 用户可以选中一张图片后进行移除，这样 SKN 文件中不再包含该文件。
- 当选中其中的一张图片后，右边预览区即显示该图片。

## 区域选择区：


- 该区域有四个值，分别对应选区的左边、顶边、右边、底边。
- 左边值不小于 0，顶边值不小于 0，右边值不大于图片的宽度，底边值不大于图片的高度。
- 该区域指定 ImageBase 选择当前图片的某个区域作为其图像源。
- 用户可以在图片预览区直接操作图片选框来大致选择图像选区。
- 用户也可以通过点击四个值旁边的 Spin 控件，对选区进行微调。

## 图片不被拉伸区：

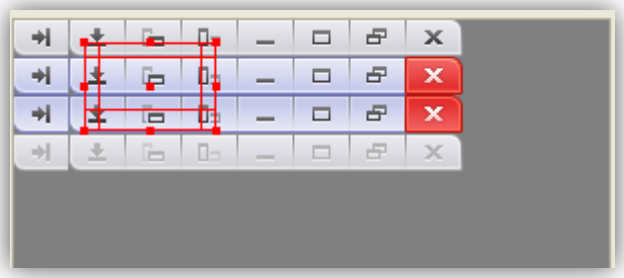



- 该区域有四个值，分别指定了图像中不能被拉伸的 4 条边的大小。分别对应左边、顶部、右边、底部。
- 用户可以通过点击四个值旁边的 Spin 控件，对不被拉伸区域进行微调。
- 在每个方向的边上与中间区域还有拉伸与平铺的互斥属性，他们用于指定 5 个缩放区域是否允许拉伸。
- 用户可以点击四个值旁边的 CheckBox 控件来设置拉伸或平铺。CheckBox 被选中表示该边属于拉伸状态，否则属于平铺状态。
- 用户也可以通过选中拉伸菜单的 Tile All 菜单项来对 5 个部分全部设置为平铺，选中 Stretch All 菜单项来对 5 个部分全部设置为拉伸。

## 透明色设置区：


- 是否选中透明色前面的 CheckBox，表示在该图片使用是否采用透明色。
- 如果该图片中不存在透明色的概念，请不要勾选该 CheckBox。这样会提高 DirectUI 绘图的效率。
- 如果勾选了 CheckBox，则需要设置透明色的 RGB 值。

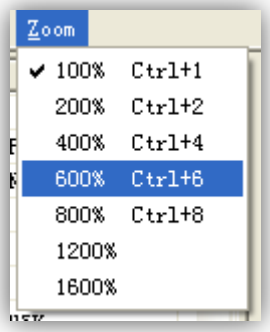
## 图像预览区：



- 该区可以对选中的图片进行显示。

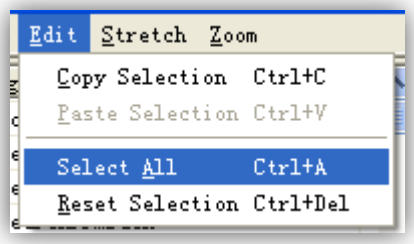
- 可以对显示的图像进行放大与缩小。
- 可以通过鼠标直接拖拽图像选框来确定选区位置与大小。
- 对于选框的拖拽的范围作了边界处理，避免用户的误操作。

## 缩放菜单：




- 可以对图像进行 100% ~ 1600%的缩放
- 图像缩放后，图像选框也跟着一起缩放，同时选框的位置与大小的调整保持以像素为单位。

## 拷贝与粘贴选区菜单：




- 拷贝一个 ImageBase 中的选区信息（包括选中的图像、选区位置与大小、不被拉伸区域的数值与拉伸与否的状态、是否透明与透明色）。
- 在另外一个 ImageBase 中执行粘贴选区，则将选区信息拷贝了过来。
- 一般在一个控件具有多个状态时需要快速的设置其 ImageBase 时采用。由于通常情况下，我们将相同控件的图像放在一个图片文件中，同时他们的选区相对靠近，其他信息又几乎一致，此时使用拷贝粘贴功能可以尽量避免重复操作，大幅度提供设置 ImageBase 的速度。

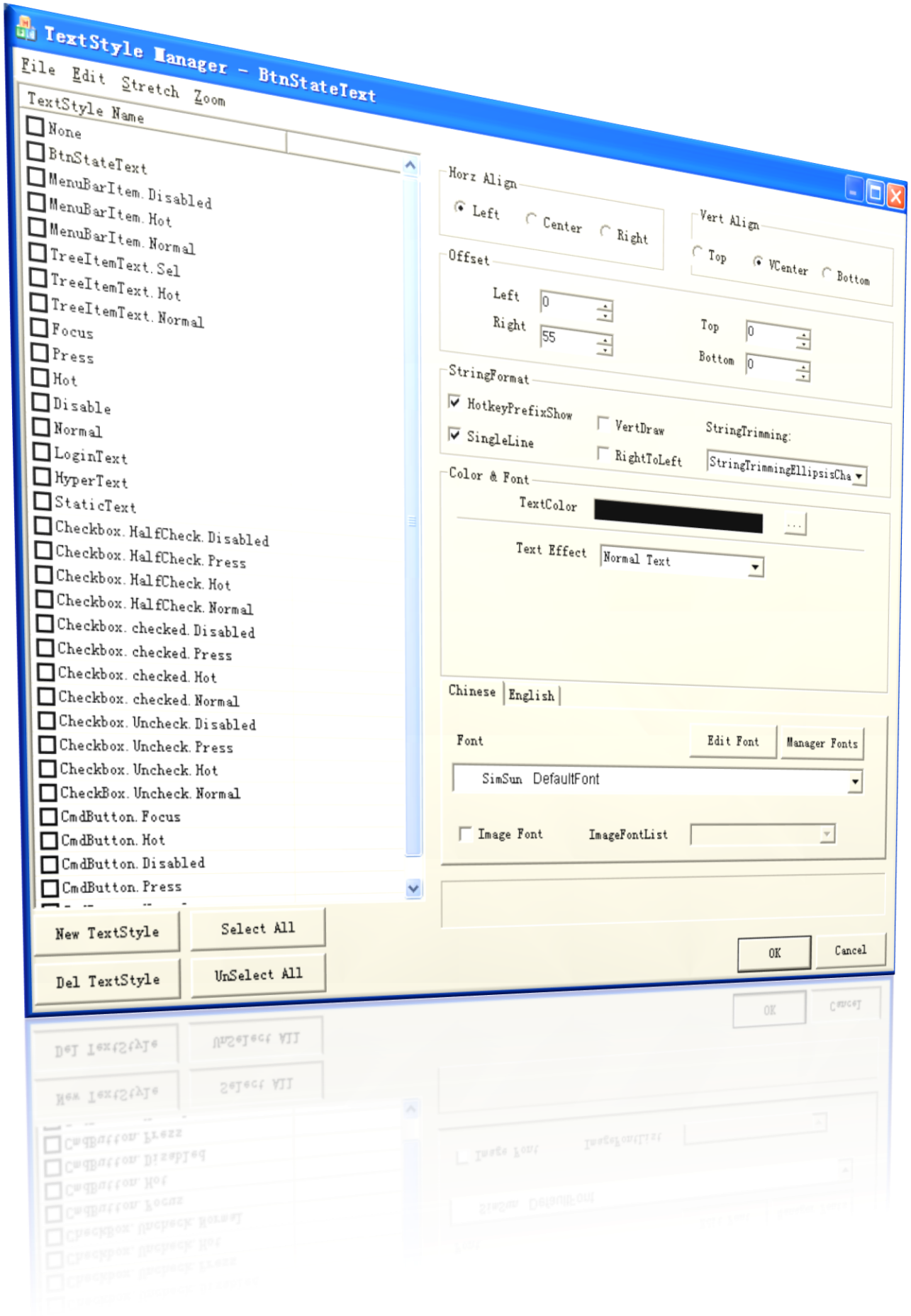
## 控制按钮区：


- 在该区包含 Reset、OK、Cancel3 个按钮
- Reset 用于初始化当前 ImageBase 的选区信息：
  - 图片选择为无
  - 选区值：left = 0, top = 0, right = 20, bottom = 20.
  - 不被拉伸区域：left = 0, top = 0, right = 0, bottom = 0

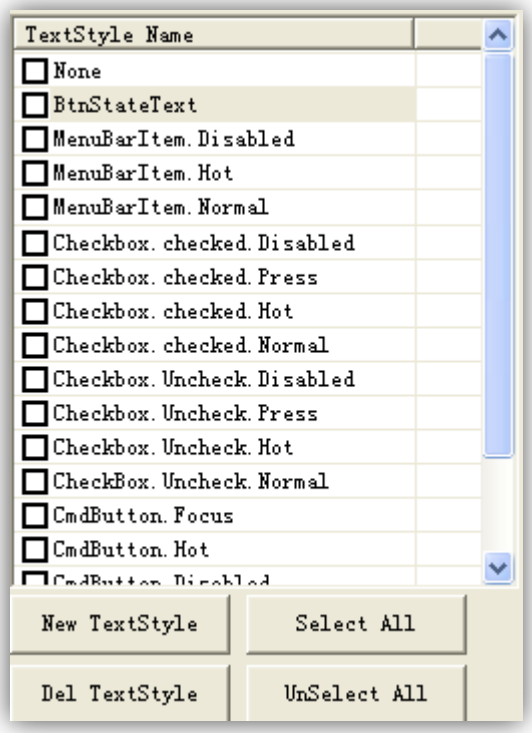
    不被拉伸区域所有边为平铺状态
  - 是否使用透明色 CheckBox 为不勾选，透明色值：Red = 255,Green = 0,Blue = 255
- OK 按钮为保证当前对 ImageBase 所做的修改，并对样式区的属性值设置为当前选中的 ImageBase。
- Cancel 按钮为保证当前对 ImageBase 所做的修改，但不会对样式区的属性值设置进行改变。
- 若要将样式区属性值设置为空，要先选中 ImageBase 列表中第一个 NONE 项，然后点击 OK 按钮。

## 文本风格设置界面

该界面分 2 大部分：TextStyle 管理区与文本风格设置区。


TextStyle Manager - BtnStateText
File Edit Stretch Zoom
TextStyle Name
None
BtnStateText
MenuBarItem.Disabled
MenuBarItem.Hot
MenuBarItem.Normal
TreeItemText.Sel
TreeItemText.Hot
TreeItemText.Normal
Focus
Press
Hot
Disable
Normal
LoginText
HyperText
StaticText
Checkbox.HalfCheck.Disabled
Checkbox.HalfCheck.Press
Checkbox.HalfCheck.Hot
Checkbox.HalfCheck.Normal
Checkbox.checked.Disabled
Checkbox.checked.Press
Checkbox.checked.Hot
Checkbox.checked.Normal
Checkbox.Uncheck.Disabled
Checkbox.Uncheck.Press
Checkbox.Uncheck.Hot
CheckBox.Uncheck.Normal
CmdButton.Focus
CmdButton.Hot
CmdButton.Disabled
CmdButton.Press
New TextStyle
Select All
Del TextStyle
UnSelect All
Horz Align
Left
Center
Right
Vert Align
Top
VCenter
Bottom
Offset
Left
0
Right
55
Top
0
Bottom
0
StringFormat
HotkeyPrefixShow
SingleLine
VertDraw
RightToLeft
StringTrimming:
StringTrimmingEllipsisCha
Color & Font
TextColor
Text Effect
Normal Text
Chinese
English
Font
Edit Font
Manager Fonts
SimSun DefaultFont
Image Font
ImageFontList
OK
Cancel

## TEXTSTYLE 管理区




- TextStyle 区管理当前 SKN 文件中所有的 TextStyle，用户可以对其进行创建、编辑与删除等操作。
- TextStyle 的命名不能重复。
- TextStyle 的命名使用英文或数字的组合，不能使用特殊字符。
- TextStyle 的命名尽量做到容易读懂。
- 通过列表前面的复选框，可以部分选择或全选，进行删除操作。

## 文本风格设置区

TextStyle 对包含文本的横向与纵向的对齐方式、横向与纵向的偏移量、字符串格式、文字效果、文字前景颜色、文字阴影颜色、多语种字体设置等信息，所以在该区域包含以下部分：

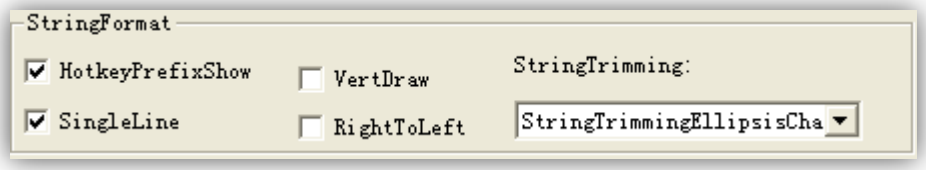
### 对齐方式：


- 横向：靠左、横向居中、靠右
- 纵向：靠上、纵向居中、靠下

## 偏移量：


- 横向偏移：left,right 数值可以对上面的横向对齐方式进行微调
- 纵向偏移：top,bottom 数值可以对上面的纵向对齐方式进行微调

## 字符串格式：




字符串格式对字符串的转义符的绘制、单行与多行、垂直绘制、从右到左绘制、字符串截断等进行了设置

- 转义符：HotkeyPrefixShow 复选框选中代表字符串中的“&”符号转义成下划线，如：&File ,DirectUI 将绘制成：File

  复选框未选中代表字符串中的“&”符号不进行转义，上面的那个例子将绘制成：&File
- 单行与多行：SingleLine 复选框选中代表字符串以单行的形式绘制，否则以多行的形式绘制。
- 垂直绘制：VertDraw 复选框选中代表以垂直方向绘制字符串。一般在纵向 TabCtrl 的 Tab 标签中的文字使用该选项。
- 从右到左绘制：RightToLeft 复选框选中代表以从右向左的方向绘制文本。一般在阿拉伯系统中用到。
- 超长后截断：当文字的长度超过控件所能显示的范围则需要对字符串进行截断，而对控件进行截断则有如下的模式：

StringTrimmingNone：不做截断，超出范围的部分由用户来处理。有可能用户看到半个字符结尾的情况。

StringTrimmingCharacter:以字符为单位的截断，但在截断后的文本中没有省略号“...”。

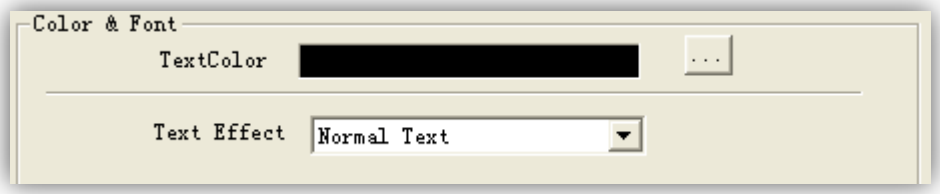
StringTrimmingWord:以单词为单位的截断，，但在截断后的文本中没有省略号“...”。

StringTrimmingEllipsisCharacter: 以字符为单位的截断，并在截断后的文本中带有省略号“...”。

StringTrimmingEllipsisWord: 以单词为单位的截断，并在截断后的文本中带有省略号“...”。

StringTrimmingEllipsisPath:以路径为单位的截断，并在截断后的文本中带有省略号“...”。

## 文字效果：




DirectUI 支持以下几种的文字效果：

- Normal：正常文本绘制，不带任何特殊效果。
- Shadowed：投影文本绘制，分前景层与投影层，用户可以分别设置 2 者的颜色 RGB 值。
- Outlined：轮廓式文本绘制，分正常文字层与轮廓层，用户可以分别设置 2 者的颜色 RGB 值。
- Shadowed On PerPixel：类似与 Vista 与 Windows7 标题栏文本的绘制方式，用户可以设置 2 者的颜色，还可以设置模糊晕层的增量值，默认为 3。

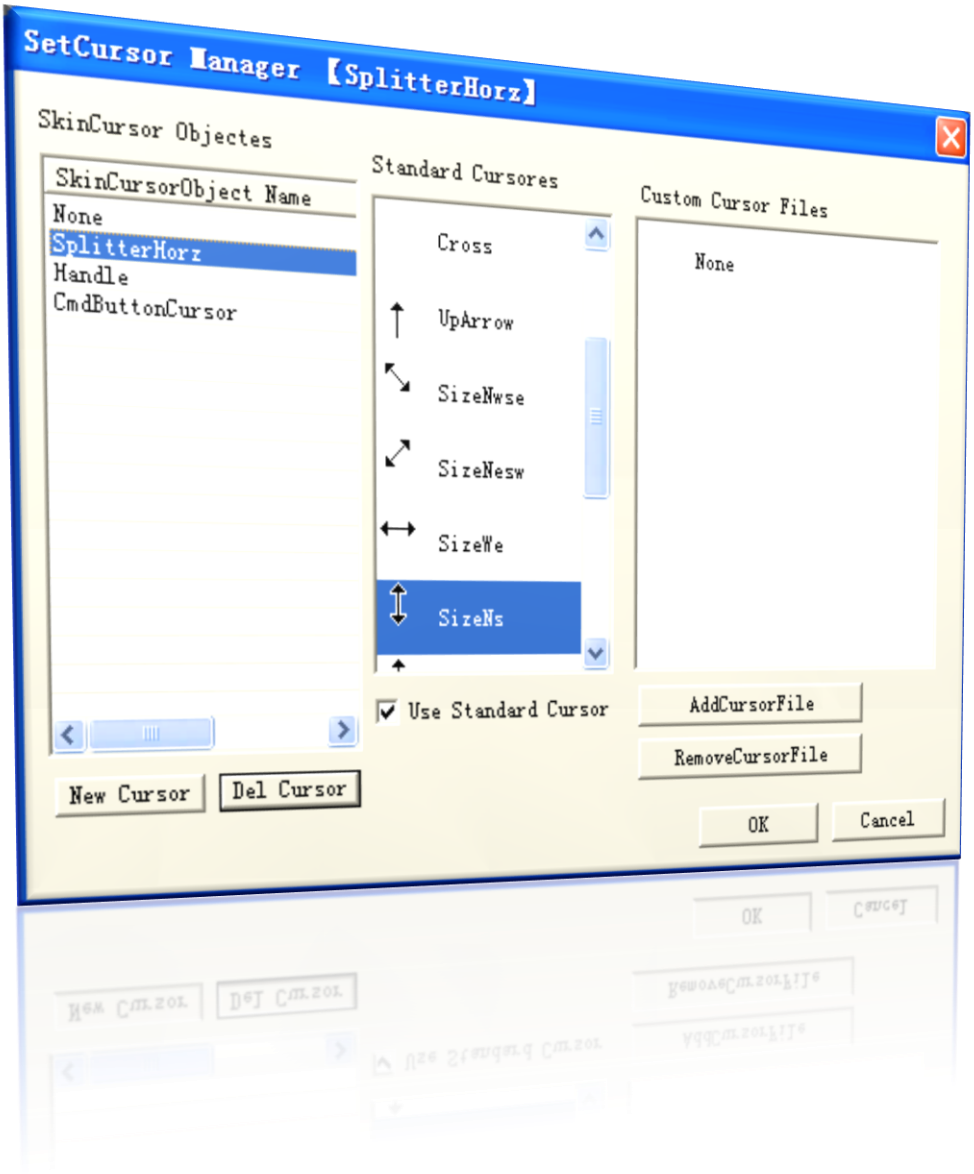
## 多语种字体：


- 由于字体与界面当前使用的语种有关系，所以 DirectUI 的文本字体信息是从属于当前语种的。
- 当前界面方案中有多少个语种，在该区就会出现多少个语种 Tab 页面，在每个 Tab 页面可以设置当前文本风格所用字体。
- 用户同时还可以创建、编辑、删除字体。
- 在某些应用场合，还需要用到图片式字体，例如 LED 数字，或特定样式的英文字符，可以创建 ImageFont。当选中 Image Font 复选框后，右边的 ImageFontList 则使能，用户可以在这里选择一个字体来使用。此时，上方的标准型字体则为禁用状态。

### 效果预览区：

该区对上面的设置进行即时的更新，使得各项配置的改变，都能及时看到最新配置的效果。

### 光标资源设置界面



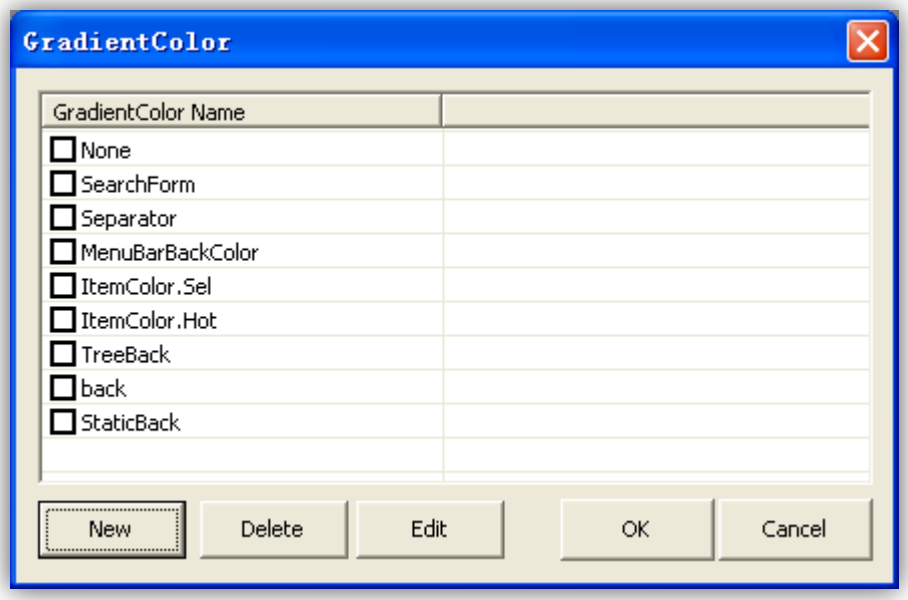

该界面包含 SkinCursor 管理区、SkinCursor 设置区。

- SkinCursor 管理区：可以新建、删除、编辑某项 SkinCursor 对象。
- SkinCursor 设置区：设置区分系统标准光标与自定义光标文件。勾选 Use Standard Cursor，则使用系统标准光标，否则使用自定义光标。
  - 系统标准光标：Windows 系统标准光标在中间的列表中进行了枚举，可以选择其中一项。
  - 自定义光标文件：点击 AddCursorFile 按钮来添加一个自定义光标文件（支持.cur,.ani 2 种格式），点击 RemoveCursorFile 按钮来移除一个自定义光标文件。

## 脚本编辑界面

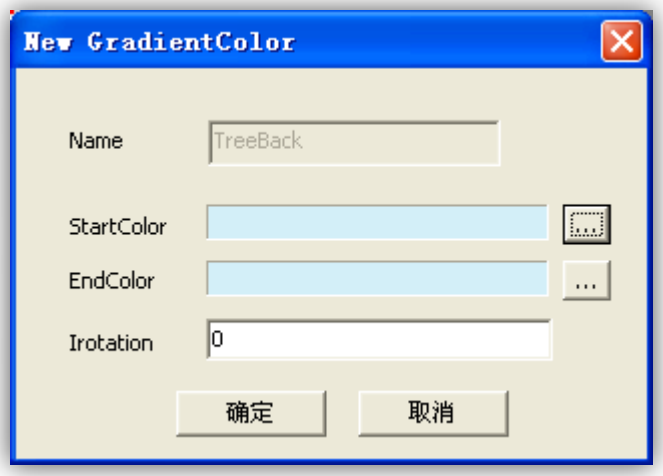
DirectUI 支持 JavaScript 脚本调用，可以在该界面写入控件的脚本代码，实现界面逻辑与业务逻辑的分离。

## 颜色管理界面




点击某个控件样式属性中的 Color，则弹出上面这个对话框。上部是 GradientColor 的列表，下部有新建、删除和编辑 GradientColor 的命令按钮。用户选中其中一项后点击 OK 按钮 ,则样式属性 Color 的值更新为选中那一项。如果选中 None 一项，点击 OK 按钮后，样式属性 Color 的值则显示为--none--。

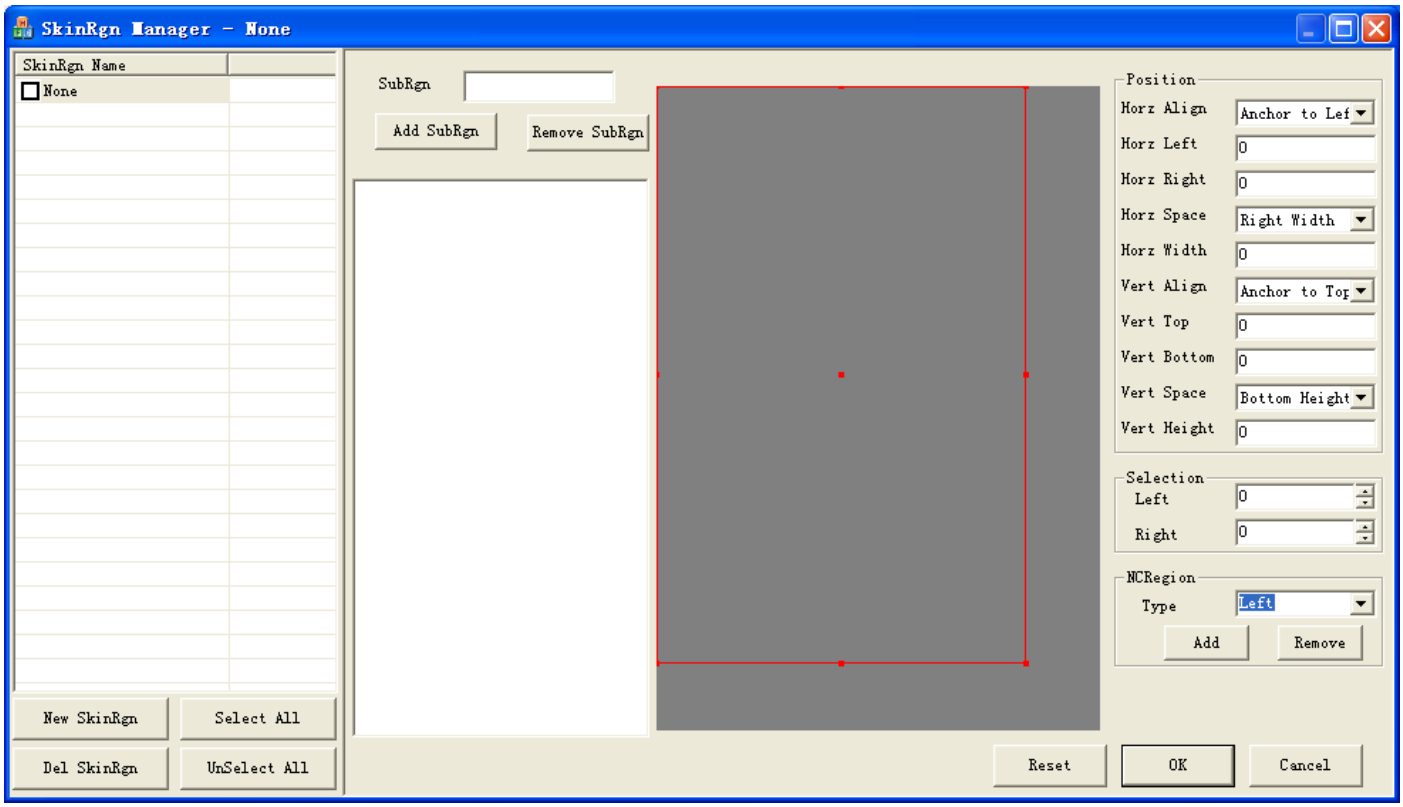
当用户选中某项后，点击 Edit 按钮，则弹出下面窗口：




- Name：显示被编辑 GradientColor 的资源名称。
- StartColor：渐变色的开始颜色。

- EndColor:渐变色的结束颜色。
- Irotation:渐变方向的角度。0°为从上到下渐变，180°为从左到右渐变。

## 区域管理界面




该界面分分 2 部分：SkinRgn 管理区和 SkinRgn 编辑区

## SKINRGN 管理区

- SkinRgn 区管理当前 SKN 文件中所有的 SkinRgn 对象，用户可以创建、编辑与删除等操作。
- SkinRgn 的命名不能重复。
- SkinRgn 的命名使用英文或数字的组合，不能使用特殊字符。
- SkinRgn 的命名尽量做到容易读懂。
- 通过列表前面的复选框，可以部分选择或全选，进行删除操作。

## SKINRGN 编辑区

该区分 3 个部分：子区域列表、中间预览区、子区域的属性

## 子区域列表

- 在一个 SkinRgn 对象中包含若干个子区域 SubRgn。
- 在 SubRgn 编辑框中填写新创建子区域的名称，然后点击 Add SubRgn 按钮，即添加一个子区域。
- 在子区域列表中选中一项，点击 Remove SubRgn 按钮，即删除一个子区域。

## 中间预览区

- 预览区显示当前控件的图像
- 控件图像上显示子区域及不被拉伸区，显示随子区域列表选中项的改变而改变。
- 在该预览区不支持鼠标拖拽选区的功能。要调整其选区位置，需要调整子区域属性区的相应的数值。

## 子区域属性区

该区域包含 3 部分：子区域布局、不被拉伸区域、非客户区拉伸类型

## 子区域布局

该区域的布局概念同控件的布局。布局父对象为当前控件。

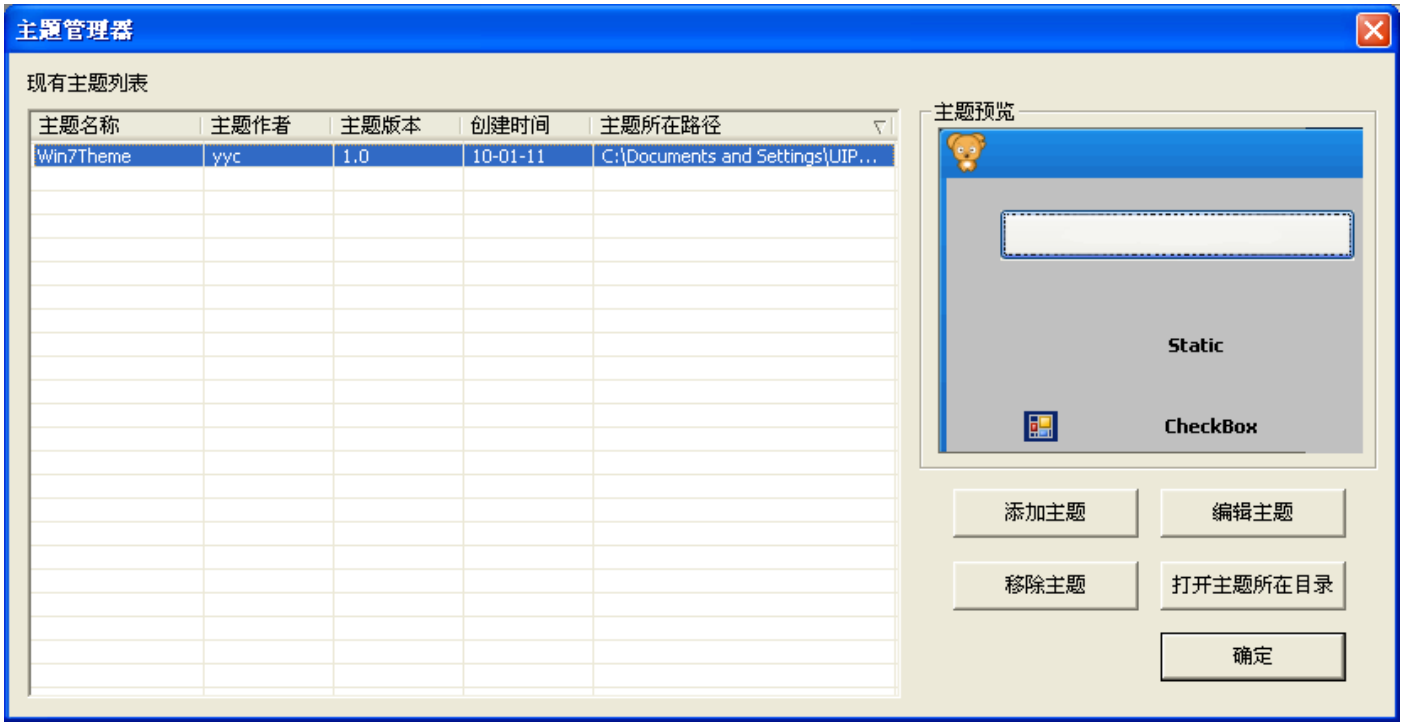
## 不被拉伸区域

- 不被拉伸区域设定 2 个方向：横向与纵向。
- 在某个子区域中，如果他是横向的，则该部分的数值则表示与左边的间距和与右边的间距。
- 在某个子区域中，如果他是纵向的，则该部分的数值则表示与顶边的间距和与底边的间距。

## 非客户区拉伸类型

- 有些子区域涉及到窗口非客户区域的拉伸，在这个地方可以进行设置。
- 非客户区域的拉伸有如下情况：Top、Left、Bottom、Right、TopLeft、TopRight、BottomLeft、BottomRight。
- 可以勾选 Use NCRegion 复选框来设置是否使用非客户区拉伸。

## 主题管理界面

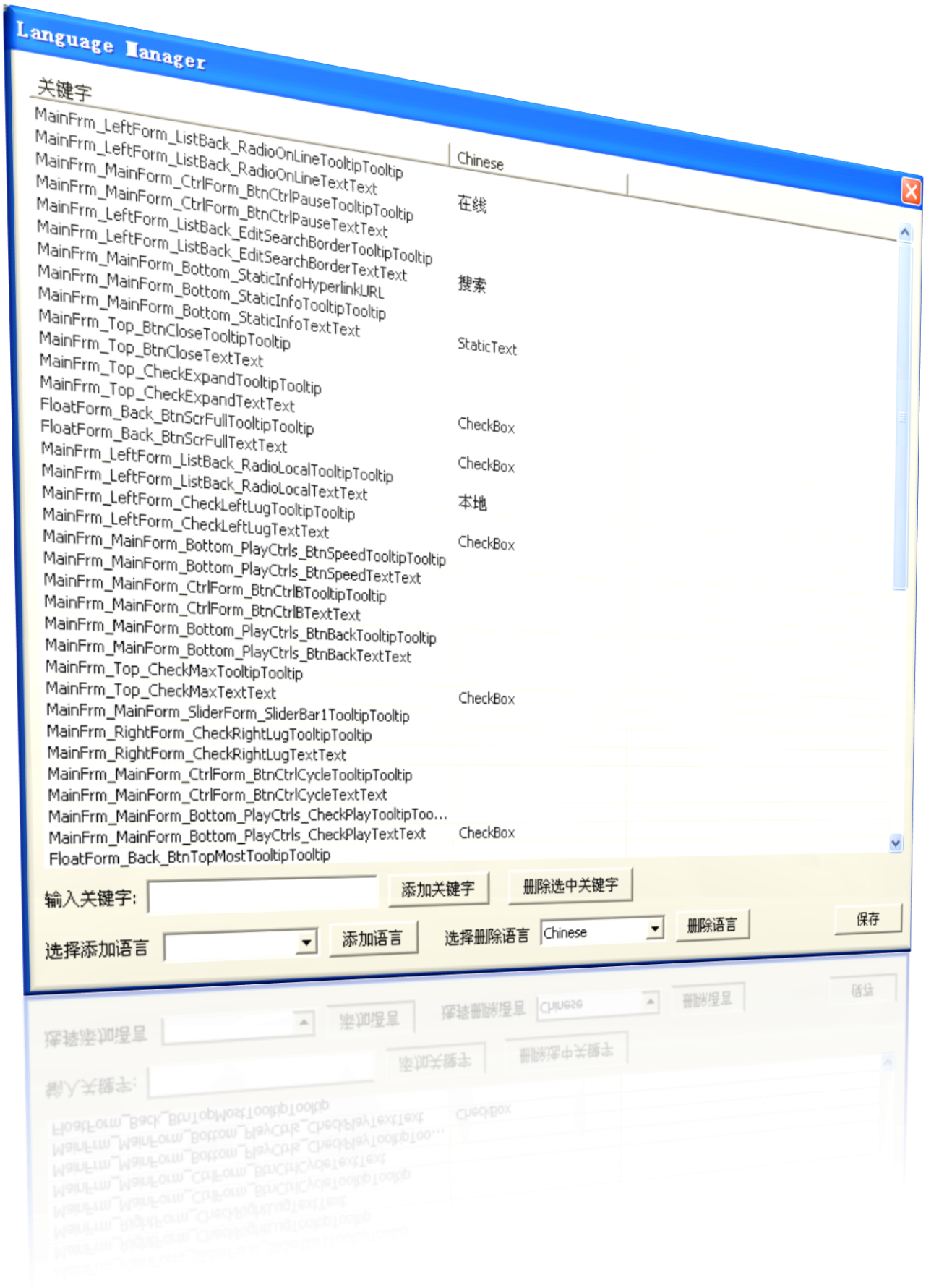



主题是 DirectUI 界面方案中很重要的一个概念。通过主题我们可以将重复的工作模板化。

该界面包括：主题列表、主题预览区、添加主题按钮、编辑主题按钮、移除主题按钮、打开主题所在目录按钮，确定按钮。

- 主题列表：列出 DirectUIBuilder 配置的所有的主题，包含作者、版本、创建时间、路径等信息。
- 主题预览区：选中列表中一个主题，预览区显示该主题的效果图。
- 添加主题：通过打开文件对话框选择磁盘中的一个主题文件，DirectUIBuilder 则记录下主题的文件路径。下次打开该列表中的主题还将存在。
- 编辑主题：选中列表中的一个主题，编辑该主题。DirectUIBuilder 将自动创建一个新的解决方案并将.dtheme 文件载入工程中，用户编辑后即保存至.dtheme 文件。
- 移除主题：选中列表中的一个主题，可以移除该主题。
- 打开主题所在目录：打开列表中选中主题所在的目录。

## 多语种管理界面


Language Manager
关键字
Chinese
MainFrm_LeftForm_ListBack_RadioOnLineTooltipTooltip
MainFrm_LeftForm_ListBack_RadioOnLineTextText
在线
MainFrm_MainForm_CtrlForm_BtnCtrlPauseTooltipTooltip
MainFrm_LeftForm_ListBack_EditSearchBorderTooltipTooltip
MainFrm_MainForm_CtrlForm_BtnCtrlPauseTextText
MainFrm_LeftForm_ListBack_EditSearchBorderTextText
搜索
MainFrm_MainForm_Bottom_StaticInfoHyperlinkURL
MainFrm_MainForm_Bottom_StaticInfoTooltipTooltip
MainFrm_MainForm_Bottom_StaticInfoTextText
MainFrm_Top_BtnCloseTooltipTooltip
StaticText
MainFrm_Top_BtnCloseTextText
MainFrm_Top_CheckExpandTooltipTooltip
MainFrm_Top_CheckExpandTextText
FloatForm_Back_BtnScrFullTooltipTooltip
CheckBox
FloatForm_Back_BtnScrFullTextText
MainFrm_LeftForm_ListBack_RadioLocalTooltipTooltip
CheckBox
MainFrm_LeftForm_ListBack_RadioLocalTextText
本地
MainFrm_LeftForm_CheckLeftLugTooltipTooltip
MainFrm_LeftForm_CheckLeftLugTextText
MainFrm_MainForm_Bottom_PlayCtrls_BtnSpeedTooltipTooltip
CheckBox
MainFrm_MainForm_Bottom_PlayCtrls_BtnSpeedTextText
MainFrm_MainForm_CtrlForm_BtnCtrlBTooltipTooltip
MainFrm_MainForm_CtrlForm_BtnCtrlBTextText
MainFrm_MainForm_Bottom_PlayCtrls_BtnBackTooltipTooltip
MainFrm_MainForm_Bottom_PlayCtrls_BtnBackTextText
MainFrm_Top_CheckMaxTooltipTooltip
MainFrm_Top_CheckMaxTextText
CheckBox
MainFrm_MainForm_SliderForm_SliderBar1TooltipTooltip
MainFrm_RightForm_CheckRightLugTooltipTooltip
MainFrm_RightForm_CheckRightLugTextText
MainFrm_MainForm_CtrlForm_BtnCtrlCycleTooltipTooltip
MainFrm_MainForm_CtrlForm_BtnCtrlCycleTextText
MainFrm_MainForm_Bottom_PlayCtrls_CheckPlayTooltipToo...
MainFrm_MainForm_Bottom_PlayCtrls_CheckPlayTextText
CheckBox
FloatForm_Back_BtnTopMostTooltipTooltip
输入关键字:
添加关键字
删除选中关键字
选择添加语言
添加语言
选择删除语言
Chinese
删除语言
保存

该界面包括关键字列表、增加关键字、删除关键字、添加语言、删除语言等功能。

- 关键字列表：

  第一列为关键字，一般有系统自动生成，名称由对象的绝对路径+属性的名称构成。

  第二列开始为语言列，一种语言一列，是对左边关键字的语言翻译。用户可以直接编辑选定关键字的某种语言的单元格。

- 增加关键字：除了系统自动生成的关键字以外，用户还可以添加自己的关键字，只要与列表中的关键字不重名即可。

  一般与界面元素没有关系，同时程序中又需要用到的文字，并且该文字又需要支持多语种的情况才需要添加用户自己的关键字。

- 删除关键字：选中列表中的关键字，点击删除选中的关键字按钮即可删除。
- 添加语言：先选中可选语言列表中语言，点击添加语言按钮，即添加了一种语言，在关键字列表中会增加一列，显示语种的名称。
- 删除语言：先选中现有语言列表，点击删除语言按钮，即删除了选定的语言，在关键字列表中会将该列删除掉。

### 设置界面方案密钥界面

### 界面向导

### 2.1.2 DSLN + DUI + SKN + DEXPORT 架构介绍

### 2.1.3 脚本介绍

### 2.1.4 加密体系介绍

### 2.1.5 多语种介绍

## 2.2 使用方法介绍

## 3.3 资源类对象

### 3.3.1 IMAGESECTION

### 3.3.2 TEXTSTYLE

### 3.3.3 CURSOR

#### 3.3.4 COLOR

## 3.4 布局接口

# 四、各控件使用介绍

## 3.1 KERNEL 控件介绍